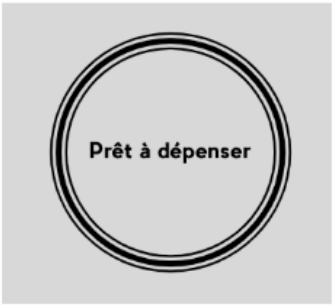


PROJET 7 : IMPLEMENTEZ UN MODELE DE SCORING

NOTE METHODOLOGIQUE

La société financière « Prêt à dépenser » propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt. Le projet consiste à développer, pour cette entreprise, un modèle de scoring de la probabilité de défaut de paiement d'un client. La décision d'accorder ou non un prêt à un client potentiel s'appuiera sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.). Par la suite, un dashboard interactif présentant les informations personnelles des clients sera développé pour permettre aux chargés de relation client de leur expliquer de façon transparente les décisions d'octroi de crédit.

Cette note méthodologique présentera :

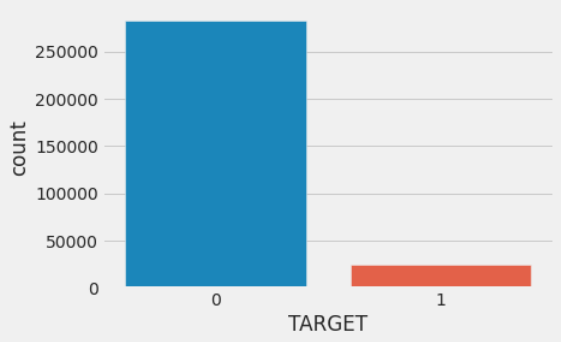
- Les données
- Méthodologie d'entraînement du modèle
- La fonction coût, l'algorithme d'optimisation et la métrique d'évaluation
- L'interprétabilité du modèle
- Les limites et les améliorations possibles

DESCRIPTION DU PROJET ET DES DONNEES

Nous utilisons une base de données de 307 511 clients provenant de la plateforme Kaggle. Celle-ci contient des informations sur l'âge, la situation familiale, le lieu de résidence, l'emploi, le revenu, les biens… Il y a 121 features. Parmi ces features, nous avons 16 variables catégorielles qu'il faudra encoder. Certaines colonnes ont des valeurs manquantes que nous allons imputer.

Le but de ce projet est de mettre en place un modèle de classification supervisée. Le jeu d'entrainement contient une colonne « TARGET » correspondant aux étiquettes à prédire : c'est une variable binaire, 0 si le client rembourse le prêt à temps) et 1 s'il a des difficultés à rembourser le prêt.

Un kernel Kaggle existant nous a permis de faciliter l'analyse des données :



Nous remarquons dans un premier temps un déséquilibre entre les deux classes à prédire : 92 % pour la classe 0 et 8 % pour la classe 1. Il sera nécessaire d'ajuster la distribution de classe de manière à avoir une répartition plus égalitaire.

L'étude des variables les plus corrélées avec la target nous permet de détecter une anomalie sur la variable « DAYS_EMPLOYED » puisqu'elle présente plus de 50 000 clients embauchés depuis environ 1000 ans. Cette anomalie semble avoir une influence sur la target donc nous remplaçons les valeurs anormales par des valeurs manquantes et créons une nouvelle colonne « DAYS_EMPLOYED_ANOM » indiquant si la valeur était anormale ou non.

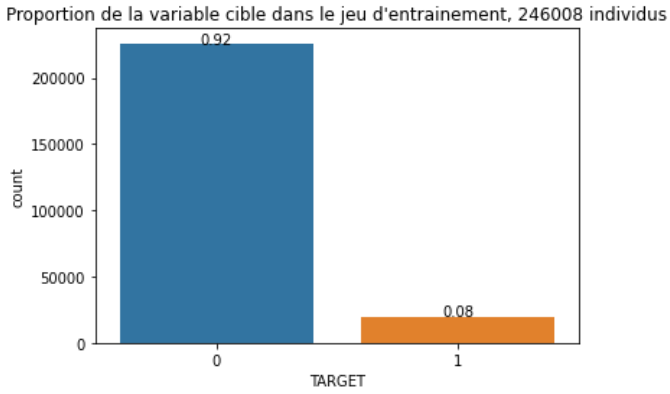
Ajouter les corrélations avec la target

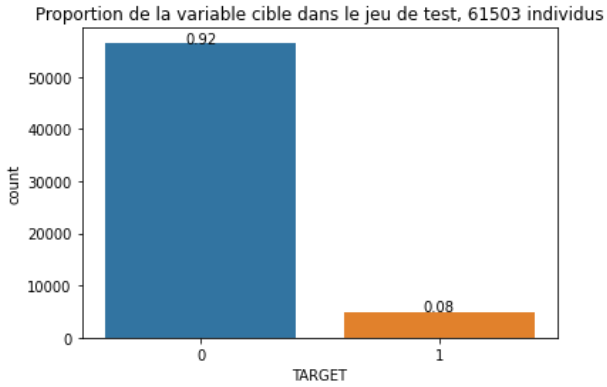
METHODOLOGIE D'ENTRAINEMENT DU MODELE

Nous devons tester notre modèle sur des données qui n'ont pas servi à l'entrainement donc nous divisions notre jeu de données en deux parties :

- un jeu d'entrainement (80% des données) qui permettra à notre modèle d'apprendre sur ces données ;
- un jeu de test (20% des données) qui nous permettra d'évaluer la performance de notre modèle.

Nous nous assurons que la proportion des classes à prédire est conservée dans chaque ensemble de données :





Pour appliquer différents traitements aux variables, nous avons créé un pipeline de prétraitement des données incluant :

- la normalisation des variables numériques. Trois approches seront testées : StandardScaler, MinMaxScaler et MaxAbsScaler ;
- l'encodage des variables catégorielles avec OneHotEncoder ;
- l'imputation des valeurs manquantes avec SimpleImputer (imputation par la médiane pour les variables numériques et par le mode pour les variables catégorielles).

Ce dernier pipeline est ensuite introduit dans un pipeline de prédiction incluant :

- une solution pour pallier le déséquilibre des classes de notre variable cible, 3 techniques seront testées :
    - RandomOverSampling (sur-échantillonnage) qui consiste à compléter les données par des copies multiples de certaines instances de la classe minoritaire ;
    - RandomUnderSampling (sous-échantillonage) qui consiste à tirer au hasard des échantillons de la classe majoritaire. Cela peut accroître la variance du classifieur et peut éventuellement éliminer des échantillons utiles ou importants ;
    - SMOTE (SyntheticMinorityOversamplingTechnique) qui consiste à sur-échantillonner la classe minoritaire en créant des exemples «synthétiques».
- une sélection d'un nombre de variables importantes pour notre variable cible avec SelectKBest ;
- plusieurs modèles de classifications.

LA FONCTION COUT, L'ALGORITHME D'OPTIMISATION ET LA METRIQUE D'EVALUATION

- **MATRICE DE CONFUSION, COURBE ROC :**

Voici la matrice de confusion associée à notre problème :

		Prédiction	
		0 (sans défaut)	1 (en défaut)
Réalité	0 (sans défaut)	Vrais négatifs	Faux positifs
	1 (en défaut)	Faux négatifs	Vrais positifs

Les faux négatifs sont les cas où la prédiction est négative, mais où la valeur réelle est positive. Ce sont des pertes réelles puisque le crédit client est accepté mais ne sera pas remboursé à temps.

Du point de vue de la banque

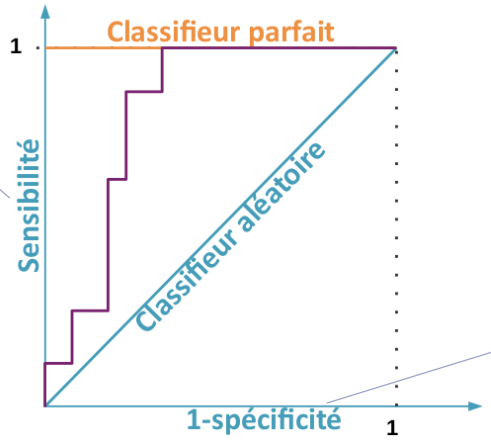
Les faux positifs sont les cas où la prédiction est positive, mais où la valeur réelle est négative. Ce sont des pertes d'opportunités puisque le crédit client est refusé à tort, alors qu'il aurait été en mesure d'être remboursé.

On cherche donc à minimiser ces erreurs de prédictions.

Notre algorithme nous retourne une probabilité que le prêt soit remboursé, pour chaque individu. Pour retourner une prédiction binaire, il faut définir un seuil de décision : si le score retourné est supérieur au seuil, alors on prédit le crédit remboursé (négatif) ; s'il est inférieur, on prédit non remboursée à temps (positif). Le seuil définit par défaut est 0,5. Nous modifierons ce seuil après avoir choisi notre modèle.

Nous choisissons d'utiliser la courbe ROC (Receiver Operating Characteristic) qui va nous permettre d'évaluer les taux de faux positifs et vrais positifs pour différents seuils de classification.

$Sensibilité = \frac{VP}{VP+FN}$ c'est le taux de vrais positifs (positifs bien identifiés), permet de réduire au maximum le taux de faux négatifs (éviter de prédire un remboursement à temps à tort).

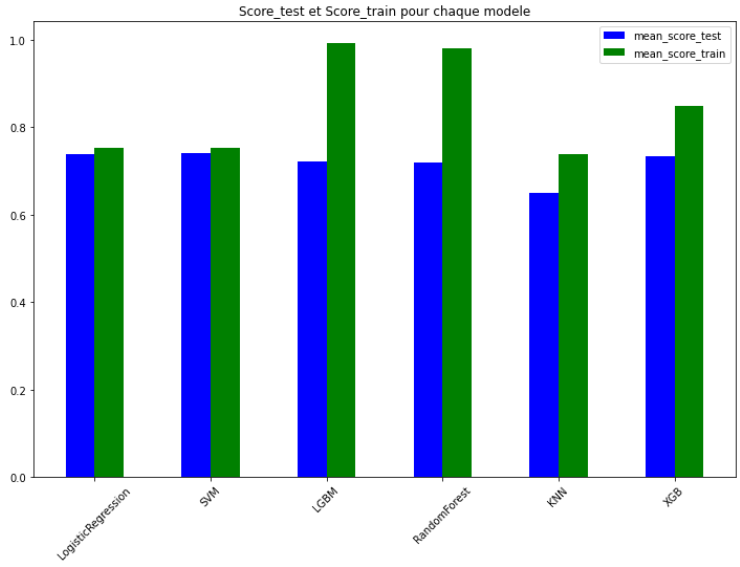


$1 - Spécificité = \frac{FP}{FP+VN}$ c'est le taux de faux positifs

Pour comparer nos différents modèles, nous essayons de maximiser un score correspondant à l'aire sous la courbe (Area Under the Curve ou AUC). Un classifieur parfait aurait un score AUROC égal à 1, tandis qu'un classifieur purement aléatoire aurait un score AUROC de 0.5.

- **RESULTATS OBTENUS AVEC LES DIFFERENTS MODELES TESTES :**

	model	mean_score_train	mean_score_test	mean_time_train	mean_time_test
0	gridlogistic	0.752420	0.739591	0.972953	0.067023
1	gridsvc	0.751906	0.740065	2.017171	0.068047
2	gridlgbm	0.992485	0.721089	3.116567	0.087681
3	gridrandomforest	1.000000	0.722695	3.993857	0.131265
4	gridknn	0.739448	0.650790	0.536391	0.167575
5	gridxgb	0.848218	0.733768	3.041840	0.074816



Nous obtenons des résultats similaires au niveau des scores AUROC avec nos différents modèles.

Les modèles *Logistic Regression Classifier* et *Support Vector Classification* donnent les meilleurs résultats au niveau des scores AUROC sur le jeu de validation. Nous continuerons avec *Logistic Regression Classifier* qui est plus rapide.

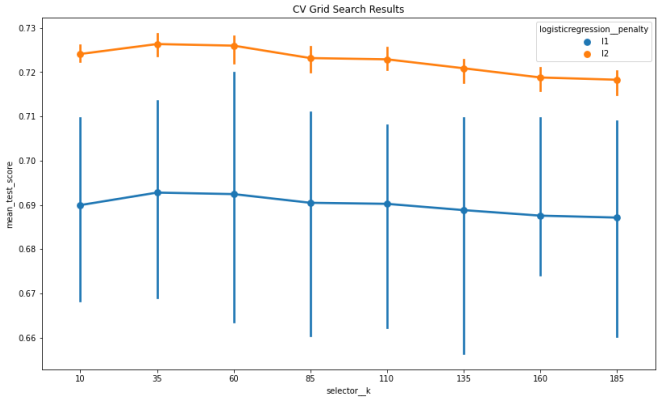
- **OPTIMISATION DU MODELE CHOISI :**

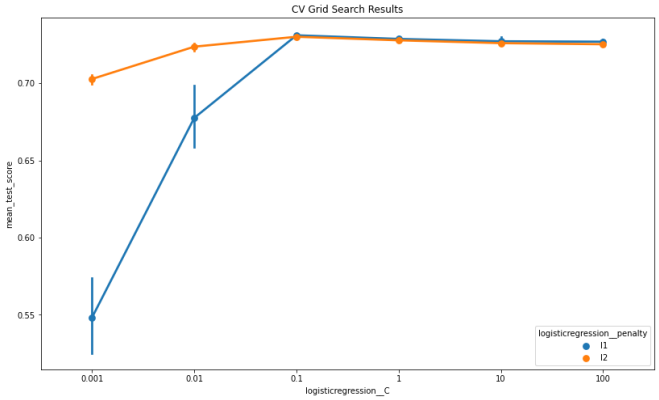
Nous cherchons à optimiser les hyperparamètres de notre modèle. Pour cela nous choisissons un ensemble de paramètres que nous souhaitons tester :

- « C » qui est l'inverse de la force de régularisation : `{0.001, 0.01, 0.1, 1, 10, 10}`
- « Penalty » qui permet de préciser la norme utilisée dans la pénalisation : `{11, 12}`
- « solver » qui est l'algorithme à utiliser dans le problème d'optimisation : `{'newton-cg', 'lbfgs', 'saga'}`
- Nous cherchons aussi à optimiser le nombre de variables optimal k à sélectionner avec SelectKBest, nous faisons varier k entre 10 et 185.

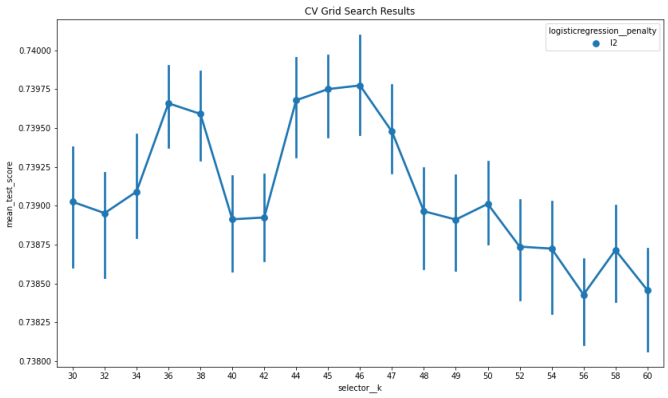
Nous utilisons Grid Search qui consiste à croiser chacune de ces hypothèses et qui va créer un modèle pour chaque combinaison de paramètres. Ces différents modèles seront entrainés et évalués en utilisant une méthode de validation croisée. Après comparaison des résultats, cela nous retournera le meilleur modèle.

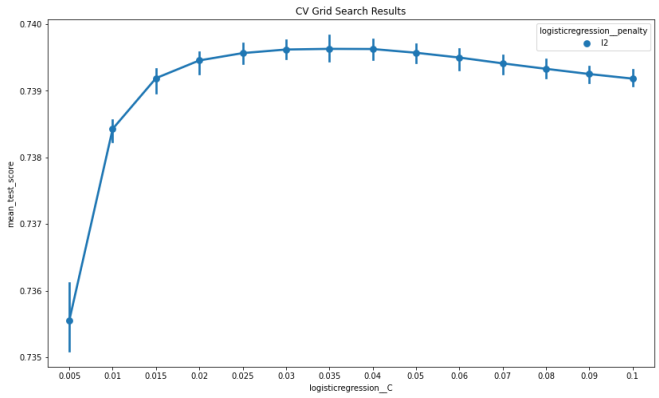
Nous présentons nos résultats graphiquement pour pouvoir affiner les valeurs de C et k :





Nous centrons notre recherche des valeurs de k autour de 50 et les valeurs de C entre 0.005 et 0.1:



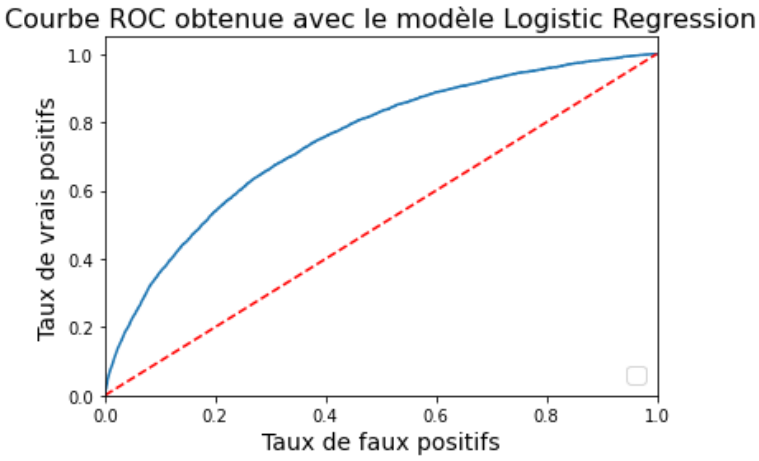


Finalement voici la grille qui nous renvoie le meilleur résultat :

```
{'logisticregression__C': 0.03,
 'logisticregression__penalty': 'l2',
 'logisticregression__solver': 'newton-cg',
 'preprocess__pipeline-1__scaler': MaxAbsScaler(),
 'sampler': RandomOverSampler(random_state=0),
 'selector__k': 46}
```

Notre optimisation ne fait pas beaucoup évoluer notre score AUROC. Nous obtenons environ 0.741 sur le jeu de validation.

- **<u>CHOIX DU SEUIL DE CLASSIFICATION :</u>**

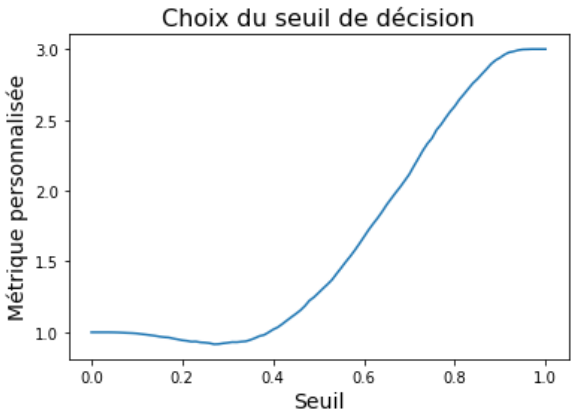


Nous allons maintenant choisir un seuil de décision pour nos prédictions. On souhaite pénaliser les faux négatifs (prêts prédits remboursés alors que non) et les faux positifs (prêts prédits non remboursés alors que oui). Nous estimons que les faux négatifs font perdre plus d'argent donc on les pénalisera davantage.

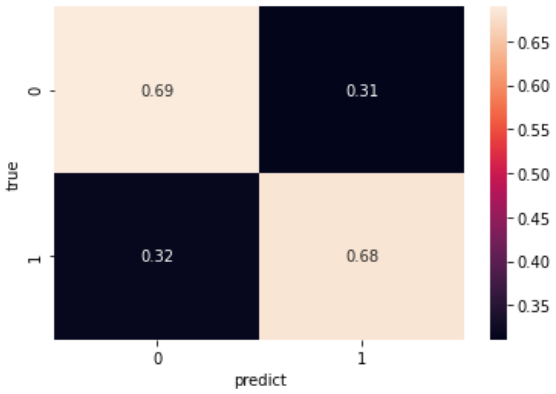
Pour des seuils compris entre 0 et 1 nous choisissons de calculer :

$$3 \times le\ taux\ de\ faux\ négatifs\ +\ le\ taux\ de\ faux\ positifs$$

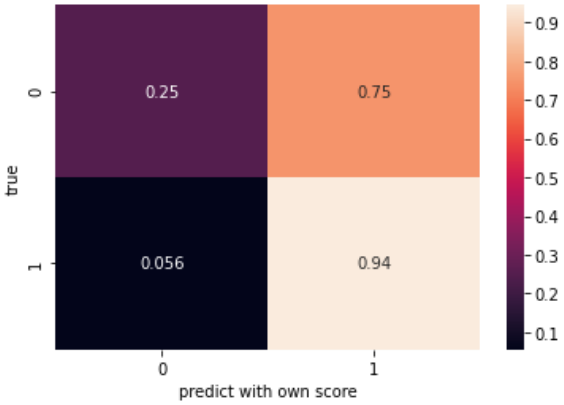
Nous voulons que cette valeur soit minimale, le seuil de décision que nous choisissons est égal à 0,27.



Nous pouvons visualiser l'impact du changement de seuil sur la matrice de confusion :



Sans la métrique personnalisée, seuil à 0.5.



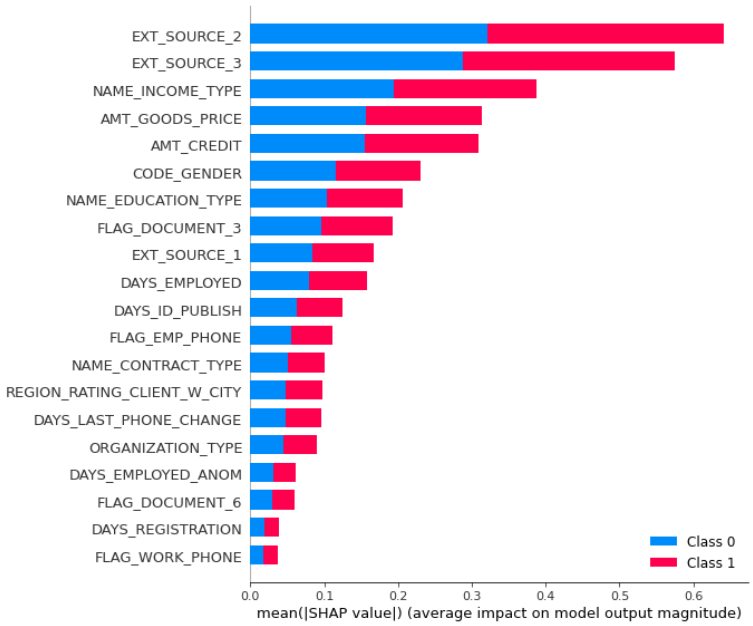
Avec la métrique personnalisée, seuil à 0.27.
Il y a beaucoup moins de faux négatifs.

INTERPRETABILITE DU MODELE

Il est nécessaire d'expliquer notre algorithme afin que les chargés de clientèles fassent confiance au modèle, comprennent la causalité de la prédiction et afin que les clients comprennent les décisions. Pour nous aider à interpréter les prédictions, nous allons utiliser le framework SHAP (SHapely Additive ExPlanations).

Les valeurs de Shapley calculent l'importance d'une variable en comparant ce qu'un modèle prédit avec et sans cette variable. Nous avons utilisé shap.KernelExplainer sur un sous-échantillon de 1 000 lignes dans notre ensemble de validation, pour accélérer le long calcul SHAP.

Nous pouvons visualiser ci-contre les 20 variables les plus importantes sur lesquelles notre modèle appuie ses prédictions :



+graphique pour une prédiction

LES LIMITES ET LES AMELIORATIONS POSSIBLES

Nous nous sommes focalisés sur une seule table de données, il y en a 7 autres mises à notre disposition que nous pourrions étudier. De plus, nos modèles pourraient être surement améliorés en travaillant avec les équipes ayant les connaissances métier. Nous pourrions ainsi :

- mieux comprendre les données avec lesquelles nous travaillons, par exemple avoir plus d'informations sur les données de type 'EXT_SOURCE' qui semblent importantes pour nos prédictions.
- avoir d'avantage de pistes pour créer des features importantes ou encore pour nettoyer nos données, gérer les outliers.
- affiner le calcul de la métrique personnalisée utilisée en estimant plus précisément le coût moyen d'un défaut de remboursement à temps, et le coût d'opportunité d'un client refusé par erreur.

Enfin, pour des raisons de temps de calcul nous avons entrainé nos modèles sur un échantillons d'individus. Augmenter cet échantillon permettrait d'améliorer l'apprentissage afin d'obtenir de meilleurs résultats sur le jeu de test.